C 13

```
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRR        RRR   MMMMMM   MMMMMM  SSS
RRR        RRR   MMMMMM   MMMMMM  SSS
RRR        RRR   MMMMMM   MMMMMM  SSS
RRR        RRR   MMM   MMM   MMM  SSS
RRR        RRR   MMM   MMM   MMM  SSS
RRR        RRR   MMM   MMM   MMM  SSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRR   RRR        MMM         MMM              SSS
RRR   RRR        MMM         MMM              SSS
RRR   RRR        MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR        RRR   MMM         MMM  SSSSSSSSSSSS
RRR        RRR   MMM         MMM  SSSSSSSSSSSSS
RRR        RRR   MMM         MMM  SSSSSSSSSSSSS
```

```
**FILE**ID**RM1CONN





RRRRRRRR    MM      MM      11        CCCCCCCC    000000    NN      NN  NN      NN
RRRRRRRR    MM      MM      11        CCCCCCCC    000000    NN      NN  NN      NN
RR      RR  MMMM  MMMM    1111      CC          00      00  NN      NN  NN      NN
RR      RR  MMMM  MMMM    1111      CC          00      00  NN      NN  NN      NN
RR      RR  MM  MM  MM      11      CC          00      00  NNNN    NN  NNNN    NN
RR      RR  MM  MM  MM      11      CC          00      00  NNNN    NN  NNNN    NN
RRRRRRRR    MM      MM      11      CC          00      00  NN  NN  NN  NN  NN  NN
RRRRRRRR    MM      MM      11      CC          00      00  NN  NN  NN  NN  NN  NN
RR  RR      MM      MM      11      CC          00      00  NN    NNNN  NN    NNNN
RR  RR      MM      MM      11      CC          00      00  NN    NNNN  NN    NNNN
RR    RR    MM      MM      11      CC          00      00  NN      NN  NN      NN      ....
RR    RR    MM      MM      11      CC          00      00  NN      NN  NN      NN      ....
RR      RR  MM      MM    111111      CCCCCCCC    000000    NN      NN  NN      NN      ....
RR      RR  MM      MM    111111      CCCCCCCC    000000    NN      NN  NN      NN      ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

RM
VO

```
RM1CONN                          SEQUENTIAL AND COMMON CONNECT          16-SEP-1984 00:44:47  VAX/VMS Macro V04-00        Page   0
Table of contents
```

```
0000            $BEGIN  RM1CONN,000,RM$RMS1,<SEQUENTIAL AND COMMON CONNECT>
0000
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000    ;
```

```
0000    ;++
0000    ; Facility: rms32
0000    ;
0000    ; Abstract:
0000    ;               routine to perform sequential-specific
0000    ;               connect processing.
0000    ;
0000    ; Environment:
0000    ;               star processor running starlet exec.
0000    ;
0000    ; Author: l f laverdure,         creation date: 5-JAN-1977
0000    ;
0000    ; Modified By:
0000    ;
0000    ;       V03-023 SHZ0026         Stephen H. Zalewski,    04-May-1984
0000    ;               If we are creating a global buffer section, specify that
0000    ;               we do an expand region to make sure it comes out of P0 space.
0000    ;
0000    ;       V03-022 JEJ0025         J E Johnson             10-Apr-1984
0000    ;               Ensure that GBH and GBD lengths are quadword aligned.
0000    ;
0000    ;       V03-021 SHZ0011         Stephen H. Zalewski,    24-Feb-1984
0000    ;               Do not initialize TRC blocks when connecting with global
0000    ;               buffers.  This was accidently left on from SHZ0010.
0000    ;
0000    ;       V03-020 SHZ0010         Stephen H. Zalewski     06-Dec-1983
0000    ;               Fix the tracing code to work with multi-threaded RMS.  This
0000    ;               is accomplished by using the interlock queue instructions.
0000    ;
0000    ;       V03-019 SHZ0009         Stephen H. Zalewski     19-Sep-1983
0000    ;               Replace call to RM$INIT_SFSB with RM$INIT_SFSB_IRB.  This is
0000    ;               to allow us to successfully stall using the irab.
0000    ;
0000    ;       V03-018 SHZ0008         Stephen H. Zalewski     10-Aug-1983
0000    ;               Bugcheck if we try to create a global buffers section
0000    ;               with global buffer count of zero.
0000    ;
0000    ;       V03-017 SHZ0007         Stephen H. Zalewski     28-Jul-1983
0000    ;               Implement cluster global buffers.
0000    ;
0000    ;       V03-016 SHZ0006         Stephen H. Zalewski     22-Jun-1983
0000    ;               Disable global buffers.
0000    ;
0000    ;       V03-015 SHZ0005         Stephen H. Zalewski     11-Apr-1983
0000    ;               Fix bug that caused a process to incorrectly map a global
0000    ;               buffer section.
0000    ;
0000    ;       V03-014 KPL0001         Peter Lieberwirth       23-Mar-1983
0000    ;               Fix v03-013 by reversing sense of branch.
0000    ;
0000    ;       V03-13  SHZ0004         Stephen H. Zalewski     21-Feb-1983
0000    ;               If XQP is being used, ignore any request for global buffers.
0000    ;               This is only a temporary restriction.
0000    ;
0000    ;       V03-012 LJA0055         Laurie J. Anderson      12-Jan-1983
0000    ;               Fill in MBF field in IRB with the value which is used
0000    ;
```

```
0000     ;       V03-011 KBT0413         Keith B. Thompson       30-Nov-1982
0000     ;               Change ifb$w_devbufsiz to ifb$l_devbufsiz
0000     ;
0000     ;       V03-010 SHZ0003         Stephen H. Zalewski,    14-Oct-1982  16:29
0000     ;               Prevent a $connect from occuring if there is no device buffer
0000     ;               or the real-time device bit is set in the device characteristics
0000     ;               field (this is also a patch in 3.2).
0000     ;
0000     ;       V03-009 SHZ0002         Stephen H. Zalewski,    10-Sep-1982  16:43
0000     ;               Remove anything and everthing to do with FRBs, SIFBs and SFDs
0000     ;               because they no longer exist.
0000     ;
0000     ;       V03-008 KBT0341         Keith B. Thompson       16-Sep-1982
0000     ;               Don't allocate multiple gbsbs when multistreaming
0000     ;
0000     ;       V03-007 SHZ0001         Stephen H. Zalewski,    1-Sep-1982  15:29
0000     ;               Modify so that global buffer section locking is now done
0000     ;               via the lock manger and the GBSB.
0000     ;
0000     ;       V03-006 KBT0299         Keith B. Thompson       24-Aug-1982
0000     ;               Reorganize psects
0000     ;
0000     ;       V03-005 KDM0002         Kathleen D. Morse       28-Jun-1982
0000     ;               Added $PCBDEF.
0000     ;
0000     ;--
0000
```

```
                            0000            .SBTTL  DECLARATIONS
                            0000
                            0000    ;
                            0000    ; Include Files:
                            0000    ;
                            0000
                            0000    ;
                            0000    ; Macros:
                            0000    ;
                            0000
                            0000            $BDBDEF
                            0000            $BLBDEF
                            0000            $CCBDEF
                            0000            $DEVDEF
                            0000            $FABDEF
                            0000            $FIBDEF
                            0000            $FWADEF
                            0000            $GBDDEF
                            0000            $GBHDEF
                            0000            $GBSBDEF
                            0000            $IMPDEF
                            0000            $IRBDEF
                            0000            $IFBDEF
                            0000            $PCBDEF
                            0000            $PRVDEF
                            0000            $PSLDEF
                            0000            $RMSDEF
                            0000            $RABDEF
                            0000            $SECDEF
                            0000            $SSDEF
                            0000            $SFSBDEF
                            0000            $TRCDEF
                            0000            $WCBDEF
                            0000
                            0000    ;
                            0000    ; Equated Symbols:
                            0000    ;
                            0000
                  00000020  0000            ROP=RAB$L_ROP*8                 ; bit offset to rop
                            0000
                            0000    ;
                            0000    ; Own Storage:
                            0000    ;
                            0000
                            0000    FAOCNTRL:
4C 58 21 24 53 4D 52 5F 00' 0000            .ASCIC  /_RMS$!XL/              ; Control string to FAO for GS name.
                        08  0000
                            0009
```

```
0009          .SBTTL  RM$CONNECT1 - SEQUENTIAL-SPECIFIC CONNECT ROUTINE
0009
0009  ;++
0009  ; RM$CONNECT
0009  ;
0009  ;       RM$CONNECT
0009  ;
0009  ;  this routine performs the following functions required
0009  ;  for connecting to sequential files:
0009  ;
0009  ;       1. perform various validity checks
0009  ;       2. if opened for block i/o allocate a lock bdb
0009  ;       3. allocate required bdb's and buffers and save count
0009  ;
0009  ;
0009  ; Calling sequence:
0009  ;
0009  ;       entered via case branch from rm$connect
0009  ;
0009  ; Input Parameters:
0009  ;
0009  ;       ap      argument list addr
0009  ;       r11     impure area addr
0009  ;       r10     ifab addr
0009  ;       r9      irab addr
0009  ;       r8      rab addr
0009  ;
0009  ; Implicit Inputs:
0009  ;
0009  ;       the contents of the rab and irab blocks
0009  ;
0009  ; Output Parameters:
0009  ;
0009  ;       r0      status
0009  ;
0009  ; Implicit Outputs:
0009  ;
0009  ;       sets various fields in the irab and ifab.
0009  ;
0009  ; Completion Codes:
0009  ;
0009  ;       the standard rms status code is set into r0 and
0009  ;       return is made to the user (not caller).
0009  ;
0009  ;       if any errors, all irab-related internal structures
0009  ;       are deallocated.
0009  ;
0009  ; Side Effects:
0009  ;
0009  ;       none
0009  ;
0009  ; note:
0009  ;
0009  ;       only 1 connected irab is allowed on a sequential file.
0009  ;       this routine assumes that this is the first irab in
0009  ;       ifab's irab chain.
0009  ;
```

```
                          0009      ;--
                          0009
                          0009      RMSCONNECT1::
                          0009
                          0009      ;
                          0009      ;  if open or create was done with bro specified (mixed block & record i/o),
                          0009      ;  check the bio rop bit and if set switch to block i/o only
                          0009      ;
                          0009
            06      E1    0009              BBC     #FAB$V_BRO,-
      0C 22 AA            000B                      IFB$B_FAC(R10),8$        ; branch if bro not set
            20      8A    000E              BICB2   #FAB$M_BIO,-
         22 AA            0010                      IFB$B_FAC(R10)           ; indicate not limited to block i/o
   04 68    2B      E1    0012              BBC     #RAB$V_BIO+ROP,(R8),8$   ; branch if bio clear in rop
            20      88    0016              BISB2   #FAB$M_BIO,-
         22 AA            0018                      IFB$B_FAC(R10)           ; switch to block i/o only
                          001A      8$:
   0C 6A    3E      E1    001A              BBC     #IFB$V_DAP, (R10), 20$   ; branch if network access
 00000000'EF        16    001E              JSB     NT$CONNECT               ; do network connect
       03 50        E8    0024              BLBS    R0, 20$                  ; continue on success
         000C       31    0027              BRW     CLN1                     ; cleanup on error
                          002A      20$:
   22 AA    05      E1    002A              BBC     #IFB$V_BIO,IFB$B_FAC(R10),-
            1B            002E                      CHKMBC                   ; branch if not block i/o
            6C      11    002F              BRB     ALLOC                    ; go to alloc if block i/o
```

```
                  0031     
                  0031     ;
                  0031     ;  error processing
                  0031     ;
                  0031     ;  record format undefined and doing record i/o processing
                  0031     ;
                  0031     
                  0031     ERRRFM:
                  0031     RMSCONN_ERRRFM::
                  0031             RMSERR  RFM
   FFC7'    30    0036     CLN1:   BSBW    RMSCCLN1                 ; deallocate irab
   FFC4'    31    0039             BRW     RMSEX_NOSTR              ; and exit
                  003C     
                  003C     ;
                  003C     ;  disk buffer size not 512, device is realtime device, or device has a zero
                  003C     ;  length device buffer.
                  003C     ;
                  003C     
                  003C     ERRDEV: RMSERR  DEV
      F3    11    0041             BRB     CLN1
                  0043     
                  0043     ;
                  0043     ;  mbc negative. (reserved for later use)
                  0043     ;
                  0043     
                  0043     ERRMBC:
                  0043             RMSERR  MBC
      EC    11    0048             BRB     CLN1
                  004A     
```

```
                            004A          .ENABL  LSB
                            004A
                            004A  ;
                            004A  ;  allocate bdbs and i/o buffers of size = blocksize (* mbc, if disk)
                            004A  ;
                            004A  ;!!!!!
                            004A  ;  \note:  might be nice to change rm$aldbuf to do split-page
                            004A  ;    (but not cross-page) allocations for unit record devices.\
                            004A  ;!!!!!
                            004A  ;
                            004A
                            004A  CHKMBC:
         55    48 AA    D0  004A          MOVL    IFB$L_DEVBUFSIZ(R10),R5 ; length of buffer
                  EC    13  004E          BEQL    ERRDEV                  ; Cannot connect if no device buffer
                  1D    E0  0050          BBS     #DEV$V_RTM,-            ;  or device is realtime device.
               E8 6A        0052                  IFB$L_PRIM_DEV(R10),ERRDEV
            54    01    D0  0054          MOVL    #1,R4                   ; mt offset for default mbf
                  0E    E1  0057          BBC     #DEV$V_FOD,-
               4B 6A        0059                  IFB$L_PRIM_DEV(R10),UNIT; branch if not disk or mt
                  18    E0  005B          BBS     #DEV$V_FOR,-
               47 6A        005D                  IFB$L_PRIM_DEV(R10),UNIT; branch if mounted foreign
            6A    1C    E1  005F          BBC     #DEV$V_RND,IFB$L_PRIM_DEV(R10),-
                  3A        0062                  ALLOC                   ; branch if not disk
                            0063
                            0063  ;
                            0063  ;  check that assumptions regarding disk buffer size are accurate
                            0063  ;
                            0063  ;  otherwise some of sequential get code won't work
                            0063  ;
                            0063
    0200 8F    55       B1  0063          CMPW    R5,#512
                  D2    12  0068          BNEQ    ERRDEV          ; it's all over if not that magic number!
                            006A
                            006A  ;
                            006A  ;  this is a connect for a disk file.
                            006A  ;
                            006A  ;  process the mbc (multi-block count) field of the rab to determine
                            006A  ;  the size of the buffers to be allocated.
                            006A  ;
                            006A
                  54    D4  006A          CLRL    R4                      ; disk offset for default mbf
         50    37 A8    98  006C          CVTBL   RAB$B_MBC(R8),R0        ; get mbc
            6A    2E    E1  0070          BBC     #IFB$V_PPF_INPUT,(R10),-
                  08        0073                  120$                    ; branch if not sys$input
            02    50    D1  0074          CMPL    R0,#2                   ; mbc at least 2?
                  1A    1E  0077          BGEQU   130$                    ; branch if yes
            50    02    D0  0079          MOVL    #2,R0                   ; set mbc=2 for sys$input
                  15    12  007C  120$:   BNEQ    130$                    ; branch if speced
 50    00000000'9F      98  007E          CVTBL   @#PIO$GB_DFMBC,R0       ; else get process default
                  0C    12  0085          BNEQ    130$                    ; branch if speced
 50    00000000'9F      98  0087          CVTBL   @#SYS$GB_DFMBC,R0       ; else get system default
                  03    12  008E          BNEQ    130$                    ; branch if speced
            50    01    D0  0090          MOVL    #1,R0                   ; else use a single block
                  AE    19  0093  130$:   BLSS    ERRMBC                  ; error if mbc negative
   55 A9    50    01    83  0095          SUBB3   #1,R0,IRB$B_MBC(R9)     ; store adjusted mbc value
            55    50    A4  009A          MULW2   R0,R5                   ; get total size of buffer
               0071     30  009D  ALLOC:  BSBW    RM$BDBALLOC             ; go allocate the buffers
```

```
1B 50   E8  00A0    331 150$:   BLBS    R0,SETNXT               ; continue on success
 FF5A'  31  00A3    332         BRW     RM$EX_NOSTR             ; exit on error.  error in
            00A6    333                                         ; rm$bdballoc returns everything
```

```
                           00A6      ;
                           00A6      ;  buffer allocation for unit record and foreign mounted devices
                           00A6      ;
                           00A6      ;  allocate a single buffer only
                           00A6      ;
                           00A6
       56    01      D0    00A6      UNIT:    MOVL    #1,R6                        ; get just one buffer/bdb
             02      E1    00A9               BBC     #DEV$V_TRM,-
          0C 6A            00AB                       IFB$L_PRIM_DEV(R10),160$; go allocate if not term
    55  0200 8F      B1    00AD               CMPW    #512,R5                      ; buffer size at least 512
             05      1B    00B2               BLEQU   160$                         ; yes, use it
    55  0200 8F      B0    00B4               MOVW    #512,R5                      ; use 512 bytes as minimum
           00A2      30    00B9      160$:    BSBW    RM$BDBALLOC_ALT              ; go allocate the buffer
             E2      11    00BC               BRB     150$                         ; do error check
                           00BE               .DSABL  LSB
                           00BE
                           00BE      ;
                           00BE      ;  perform remaining stream setup
                           00BE      ;
                           00BE
    3C A9    54      D0    00BE      SETNXT:  MOVL    R4,IRB$L_NXTBDB(R9)          ; set next bdb for seqxfr
                           00C2
                           00C2      ;
                           00C2      ;  position file for stream at beginning of file
                           00C2      ;  unless eof bit set in ifab or rop
                           00C2      ;
                           00C2
             1B      E0    00C2               BBS     #DEV$V_FOR,-
          39 6A            00C4                       IFB$L_PRIM_DEV(R10),65$ ; leave positioned at blk 0;
                           00C6                                                    ; if non-file structured
          40 A9      D6    00C6               INCL    IRB$L_NRP_VBN(R9)            ; assume at beginning of file
    04 6A    21      E0    00C9               BBS     #IFB$V_EOF,(R10),20$         ; branch if position to eof flag set
    0A 68    28      E1    00CD               BBC     #RAB$V_EOF+ROP,(R8),30$      ; branch if eof not set in rop either
                           00D1
                           00D1      ;
                           00D1      ;  copy the eof position to the next record pointer context
                           00D1      ;
                           00D1
 40 A9    74 AA      D0    00D1      20$:     MOVL    IFB$L_EBK(R10),IRB$L_NRP_VBN(R9); these better be zero
 44 A9    5C AA      B0    00D6               MOVW    IFB$W_FFB(R10),IRB$W_NRP_OFF(R9); for unit record devices
                           00DB
                           00DB      ;
                           00DB      ;  check for positioned at or past eof unless unit record
                           00DB      ;
                           00DB
                           00DB      30$:
                           00DB               ASSUME  DEV$V_REC EQ 0
          14 6A      E8    00DB               BLBS    IFB$L_PRIM_DEV(R10),50$ ; branch if unit record
          40 A9      D1    00DE               CMPL    IRB$L_NRP_VBN(R9),-
          74 AA            00E1                       IFB$L_EBK(R10)               ; nrp past eof?
             0D      1F    00E3               BLSSU   50$                          ; branch if not
             07      1A    00E5               BGTRU   40$                          ; branch if definite yes
                           00E7
                           00E7      ;
                           00E7      ;  nrp vbn = eof vbn
                           00E7      ;  must check byte in block to determine if at eof
                           00E7      ;
```

```
                               00E7
          5C AA   44 A9   B1   00E7                CMPW    IRB$W_NRP_OFF(R9),IFB$W_FFB(R10)
                     04   1F   00EC                BLSSU   50$                         ; branch if nrp < eof
                               00EE       40$:     SSB     #IRB$V_EOF,(R9)             ; set the eof flag
             01   54 A9   91   00F2       50$:     CMPB    IRB$B_BCNT(R9),#1           ; just 1 buffer allocated?
                     04   1B   00F6                BLEQU   60$                         ; branch if yes
                               00F8                SSB     #IRB$V_RAHWBH,(R9)          ; enable read ahead & write behind
                  FF01'   31   00FC       60$:     BRW     RM$EXSUC                    ; exit with success
                               00FF
                               00FF       ;
                               00FF       ;     maintains eof context on foreign devices
                               00FF       ;
                               00FF
             F9 6A   05   E1   00FF       65$:     BBC     #DEV$V_SQD,IFB$L_PRIM_DEV(R10),60$; branch if not magtape
             F5 6A   21   E1   0103                BBC     #IFB$V_EOF,(R10),60$        ; if not at eof, no problem
                               0107                SSB     #IRB$V_EOF,(R9)             ; set irab eof bit
                     EF   11   010B                BRB     60$                         ; return to mainline
```

```
                          010D
                          010D  ;++
                          010D  ;
                          010D  ;     subroutine to allocate bdbs and buffers.  code to lock buffers in working
                          010D  ;     set remains no-op'd at time of release 2.  it is felt at this time that
                          010D  ;     locking buffers in the working set when the multi-buffer count is positive
                          010D  ;     will probably cause problems with existing programs because in fact, rms
                          010D  ;     has not been locking them at all.  if this is implemented in a future
                          010D  ;     release, the cleanest technique would seem to be the addition of yet
                          010D  ;     another (sigh...) rop bit in the rab as input to the $connect operation
                          010D  ;     to specifically request rms to lock the buffers.  the current behavior
                          010D  ;     of using absolute value of the mbf field or default counts will continue.
                          010D  ;
                          010D  ;     if this is for magtape with truncate access, only 1 buffer is allocated
                          010D  ;
                          010D  ;     entry at rm$bdballoc_alt is used when buffer size is already specified in
                          010D  ;     r6.  see additional comments there.
                          010D  ;
                          010D  ;
                          010D  ;     inputs:
                          010D  ;
                          010D  ;          r11                 impure area address
                          010D  ;          r10                 ifab address
                          010D  ;          r9                  irab address
                          010D  ;          r8                  rab address
                          010D  ;          r5                  size of buffers to allocate, in bytes.
                          010D  ;          r4                  index for defaults, values as follows:
                          010D  ;
                          010D  ;                                      0 - sequential disk file default
                          010D  ;                                      1 - magtape default
                          010D  ;                                      2 - unit record default
                          010D  ;                                      3 - relative file default
                          010D  ;                                      4 - indexed file default
                          010D  ;                                      5 - hashed file default
                          010D  ;
                          010D  ;          rab$b_mbf           explicit # of buffers
                          010D  ;
                          010D  ;     outputs:
                          010D  ;
                          010D  ;          r0                  status code
                          010D  ;          r1-r6               destroyed
                          010D  ;          r4                  address of last bdb allocated
                          010D  ;          irb$b_bcnt          # of buffers allocated - updated only if r9 nonzero.
                          010D  ;
                          010D  ;          allocation failure when called from connect (r9 nonzero) will
                          010D  ;          return all allocated buffers, bdb's, bcb's, and the irab.
                          010D  ;
                          010D  ;--
                          010D
             55     7C    010D  BLKALL: CLRQ    R5                        ; this will get lock bdb only
             4D     11    010F          BRB     RM$BDBALLOC_ALT           ; extended branch
                          0111  RM$BDBALLOC::
       22 AA     05 E0    0111          BBS     #IFB$V_BIO,IFB$B_FAC(R10),-
                 F7       0115                  BLKALL                    ; block i/o then just do bdb
       56    36 A8  98    0116          CVTBL   RAB$B_MBF(R8),R6          ; get number of buffers
                 1F  12   011A          BNEQ    10$                       ; branch if specified
 56  00000000'9F44   98   011C          CVTBL   @#PIO$GB_DFMBFSDK[R4],R6; else, pick up process default
```

```
              15    12  0124           BNEQ    10$                          ; branch if specified
 56  00000000'9F44  98  0126           CVTBL   @#SYS$GB_DFMBFSDK[R4],R6; else, pick up system default
              08    12  012E           BNEQ    10$                          ; branch if specified
      56      01    D0  0130           MOVL    #1,R6                        ; else use 1 buffer
                        0133
                        0133    ;
                        0133    ;  if read ahead or write behind spec'd, then need two buffers
                        0133    ;
                        0133
                        0133           ASSUME  <<RAB$M_RAH!RAB$M_WBH>&^XFFFF00FF> EQ 0
   05 A8      06    B3  0133           BITW    #<RAB$M_RAH!RAB$M_WBH>@-8,RAB$B_ROP1(R8)
                        0137                                                ; either rab or wbh spec'd?
                        0137
              02    13  0137           BEQL    10$                          ; eql don't want rah/wbh
              56    D6  0139           INCL    R6                           ; need min two buffs
              56    D5  013B    10$:   TSTL    R6
              03    14  013D           BGTR    20$                          ; if pos, then ok
      56      56    CE  013F           MNEGL   R6,R6                        ; otherwise make it positive
                        0142    20$:
   5C A9      56    90  0142           MOVB    R6,IRB$B_MBF(R9)             ; Save MBF value used
   07 6A      36    E1  0146           BBC     #IFB$V_TEF,(R10),40$         ; branch if no truncate access
              05    E1  014A           BBC     #DEV$V_SQD,-
      03 6A             014C                   IFB$L_PRIM_DEV(R10),40$ ; branch if not magtape
      56      01    D0  014E           MOVL    #1,R6                        ; allocate 1 buffer
                        0151    40$:
                        0151
                        0151    ;
                        0151    ;  since we can't get good indexed defaults any other way
                        0151    ;  alter r6 here. indexed files require at least 2 bdb's and buffer's
                        0151    ;  so if absolute value of r6 is 1, then need to change it
                        0151    ;
                        0151
      54      04    D1  0151           CMPL    #4,R4                        ; see if indexed
              08    12  0154           BNEQ    80$                          ; if not branch
      02      56    D1  0156           CMPL    R6,#2                        ; at least 2 buffers spec'd?
              03    1E  0159           BGEQU   80$                          ; ok if greater than or equal
      56      02    D0  015B           MOVL    #2,R6                        ; use 2 otherwise
                        015E    80$:
                        015E
                        015E    ;
                        015E    ;  alternate entry point for number of buffers already specified in r6.
                        015E    ;
                        015E    ;  if r9 is zero, then irb$b_bcnt is not filled in.  this entry point is
                        015E    ;  for unit record and foreign devices to allocate a single buffer not
                        015E    ;  using the mbf or defaults.  extend and display will use this to allocate
                        015E    ;  buffers when no streams are connected (relative or isam only).
                        015E    ;
                        015E    ;  inputs:
                        015E    ;
                        015E    ;       r6                      number of buffers to
                        015E    ;                               allocate.  0 causes only one buffer to
                        015E    ;                               be allocate and bypasses potential allocation
                        015E    ;                               of lock bdb for relative and isam orgs.
                        015E    ;
                        015E    ;       ifb$v_wrtacc            if set, then allocate a lock bdb also for
                        015E    ;                               relative and isam files if low word r6 non zero
                        015E    ;
```

```
                              015E  ;       ifb$l_sfsb_ptr          if non-zero, file is shared and a bcb is
                              015E  ;                               allocated for each bdb.
                              015E  ;
                              015E  ;  outputs:
                              015E  ;
                              015E  ;       bdb's are linked into the end of the ifab bdb list.
                              015E  ;
                              015E
                              015E  RMSBDBALLOC_ALT::
               7E   D4        015E          CLRL    -(SP)                   ; init buffer counter
      04 22 AA   05   E1      0160          BBC     #IFB$V_BIO, IFB$B_FAC(R10), AGAIN ; Br if not block i/o.
                              0165          SSB     #IFB$V_NORECLK, (R10)   ; Make sure noreclk is set for bio.
                              0169  AGAIN:
               6E   D6        0169          INCL    (SP)                    ; count the buffer
            FE92'   30        016B          BSBW    RM$ALDBUF               ; allocate the buffer
            23 50   E9        016E          BLBC    R0,DECR_BCNT            ; get out on error
         06 6A 33   E0        0171          BBS     #IFB$V_NORECLK, (R10), 10$ ; branch if no record locking.
            FE88'   30        0175          BSBW    RM$ALBLB                ; Allocate a BLB.
            2E 50   E9        0178          BLBC    R0,GIVEBACK             ; branch if error on getting bcb
               55   D5        017B  10$:    TSTL    R5                      ; was buffer allocated?
               04   13        017D          BEQL    20$                     ; EQL then not, so don't count it.
          0084 CA   B6        017F          INCW    IFB$W_AVLCL(R10)        ; note buffer allocated.
            E3 56   F5        0183  20$:    SOBGTR  R6,AGAIN                ; decrement counter, go again
                              0186                                          ; if still positive
               0C   19        0186          BLSS    DECR_BCNT               ; this was last pass to alloc
                              0188                                          ; just lock bdb so decr bcnt
                              0188                                          ; so it only counts buffers
                              0188
                              0188  ;
                              0188  ; At this point the required number of buffers and bdbs, and blbs (if shared)
                              0188  ; have been allocated.  Allocate a lock blb if record locking is being done.
                              0188  ;
                              0188
         0A 6A 33   E0        0188          BBS     #IFB$V_NORECLK,(R10),EXIT ; done if no locking.
            FE71'   30        018C          BSBW    RM$ALBLB                ; Allocate a lock BLB.
            24 50   E8        018F          BLBS    R0, CHKGBL              ; Check out global buffers.
               02   11        0192          BRB     EXIT                    ; Exit on error from alblb.
                              0194  DECR_BCNT:
                              0194
                              0194  ;
                              0194  ; come here on error and
                              0194  ; last pass to get count right
                              0194  ;
                              0194
               6E   D7        0194          DECL    (SP)                    ;
                              0196  EXIT:
               02   BA        0196          POPR    #^M<R1>                 ; get buffer count off stack
               59   D5        0198          TSTL    R9                      ; is there an irab?
               07   13        019A          BEQL    10$                     ; no, then don't update bcnt
                              019C                                          ; and exit (caller checks error)
         54 A9 51   90        019C          MOVB    R1,IRB$B_BCNT(R9)       ; store count of buffers
            01 50   E9        01A0          BLBC    R0,20$                  ; error on allocation
                              01A3                                          ; clean up buffers allocated
                              01A3                                          ; and get rid of irab
                              01A3                                          ; r9 nonzero means this was
                              01A3                                          ; called on a connect
                    05        01A3  10$:    RSB                             ; and exit routine
          50   DD  01A4   20$:       PUSHL   R0                         ; save status
       FE57'   31  01A6              BRW     RMSCOMCLNUP                ; and branch to cleanup
                   01A9
                   01A9   ;
                   01A9   ; we couldn't get a blb for some reason (e.g., not enough space left).
                   01A9   ; therefore, we must return the bdb we just got.
                   01A9   ;
                   01A9
                   01A9   GIVEBACK:
          50   DD  01A9              PUSHL   R0                         ; save status code
   54  44 AA   D0  01AB              MOVL    IFB$L_BDB_BLNK(R10),R4     ; get back link because
                   01AF                                                 ; aldbuf calls albdb which
                   01AF                                                 ; links them at end of list
       FE4E'   30  01AF              BSBW    RM$RETBDB                  ; deallocates bdb @r4
          01   BA  01B2              POPR    #^M<R0>                    ; restore status code
          DE   11  01B4              BRB     DECR_BCNT                  ; fix count and exit
                   01B6
```

```
                      01B6
                      01B6   ;
                      01B6   ; Code to allocate global buffers if desired, and initialize if necessary.
                      01B6   ;
                      01B6
                      01B6   CHKGBL:
        78 AA   D5    01B6            TSTL    IFB$L_SFSB_PTR(R10)     ; If file is not being shared,
           DB   13    01B9            BEQL    EXIT                    ;  then don't bother with global buffers
      0088 CA   D5    01BB            TSTL    IFB$L_GBH_PTR(R10)      ; If we already have global buffers
           03   13    01BF            BEQL    1$                      ;  then multi-streaming.
         00A5   31    01C1            BRW     MAP_IT                  ; Go map the section.
                      01C4   1$:      ASSUME  IMP$V_IIOS EQ 0
        CF 6B   E9    01C4            BLBC    (R11), EXIT             ; Just use local if this is ppf.
     CB 6B      05 E0 01C7            BBS     #IMP$V_NOPOBUFS,(R11),EXIT ; Use local if P0 off limits.
           59   D5    01CB            TSTL    R9                      ; Is irab present?
           C7   13    01CD            BEQL    EXIT                    ; No, then just use local.
     56 3C AB   D0    01CF            MOVL    RAB$L_FAB(R8), R6       ; Get address of FAB.
     56 48 A6   32    01D3            CVTWL   FAB$W_GBC(R6), R6       ; Get gbl buffer count.
           BD   13    01D7            BEQL    EXIT                    ; None desired, so exit.
        FE24'   30    01D9            BSBW    RM$INIT_GBSB            ; Allocate GBSB and get EX lock.
        B7 50   E9    01DC            BLBC    R0,EXIT                 ; Exit if lock failed.
     54 7C AA   D0    01DF            MOVL    IFB$L_GBSB_PTR(R10),R4  ; Get address of GBSB from IFB.
     52 38 A4   D0    01E3            MOVL    GBSB$L_GS_SIZE(R4),R2   ; Are global buffers already in use?
           54   13    01E7            BEQL    CHK_GBC                 ; No, branch to validate GBC.
         007D   31    01E9            BRW     MAP_IT                  ; Yes, go use them.
                      01EC
  54   009C CA  D0    01EC   ERL0:    MOVL    IFB$L_BLBBLNK(R10), R4  ; Get addr of BLB.
        FE0C'   30    01F1            BSBW    RM$RETBLB               ; Return it.
  54   009C CA  D0    01F4   ERL1:    MOVL    IFB$L_BLBBLNK(R10), R4  ; Get BLB addr.
        FE04'   30    01F9            BSBW    RM$RETBLB               ; Give back.
     54 44 AA   D0    01FC   ERL4:    MOVL    IFB$L_BDB_BLNK(R10), R4 ; Get address of GBPB just alloc'd.
        FDFD'   30    0200            BSBW    RM$RETGBPB              ; Give it back.
     54 44 AA   D0    0203   ERL3:    MOVL    IFB$L_BDB_BLNK(R10), R4 ; Get address of GBPB just alloc'd.
        FDF6'   30    0207            BSBW    RM$RETGBPB              ; Give it back.
  54   009C CA  D0    020A   ERL2:    MOVL    IFB$L_BLBBLNK(R10), R4  ; Get addr of a BLB (lock BLB).
        FDEE'   30    020F            BSBW    RM$RETBLB               ; Give it back.
        FDEB'   30    0212            BSBW    RM$RLS_GBSB             ; Dequeue the lock we had on the GBSB
           01   BA    0215            POPR    #^M<R0>                 ; Restore error code.
         FF7C   31    0217            BRW     EXIT                    ; Go finish up.
                      021A
                      021A   ALBLBERR:
     5E    08   C0    021A            ADDL2   #8, SP                  ; Clean off stack.
           50   DD    021D            PUSHL   R0                      ; Save error code.
           D3   11    021F            BRB     ERL1                    ; Br and finish up.
                      0221   ALBLBERR1:
     5E    08   C0    0221            ADDL2   #8, SP                  ; Clean off stack.
           50   DD    0224            PUSHL   R0                      ; Save error code.
           D4   11    0226            BRB     ERL4                    ; Br and finish up.
                      0228   ALGBPERR:
     5E    08   C0    0228            ADDL2   #8, SP                  ; Clean off stack.
           50   DD    022B            PUSHL   R0                      ; Save error code.
           DB   11    022D            BRB     ERL2                    ; Br to give back lock BLB.
                      022F   ALGBPERR1:
     5E    08   C0    022F            ADDL2   #8, SP                  ; Clean off stack.
           50   DD    0232            PUSHL   R0                      ; Save error code.
           CD   11    0234            BRB     ERL3                    ; Br to give back one gbpb.
                      0236   BAD_GBC:
                              0236          RMSERR  GBC,-(SP)                       ; Note error.
                    CD   11   023B          BRB     ERL2                            ; Give back lock BLB.
                              023D
                              023D          ASSUME  <<GBH$C_BLN/8>*8> EQ GBH$C_BLN  ; Check for quadword alignment
                              023D          ASSUME  <<GBD$C_BLN/8>*8> EQ GBD$C_BLN  ; in GBD and GBH sections
                              023D
                              023D  CHK_GBC:
               52   56   D0   023D          MOVL    R6, R2                          ; Save number of buffers desired.
                    F4   19   0240          BLSS    BAD_GBC                         ; Only positive values allowed.
          51   52   55   C5   0242          MULL3   R5, R2, R1                      ; Total buffer bytes into R1.
               52   28   C4   0246          MULL2   #GBD$C_BLN, R2                  ; R2 is now descriptor bytes.
               52   51   C0   0249          ADDL2   R1, R2                          ; Sum of desc and buffers.
     52   00000058  8F   C0   024C          ADDL2   #GBH$C_BLN, R2                  ; Plus size of header area.
     52   000001FF  8F   C0   0253          ADDL    #511,R2                         ; Round up to even pages.
     52   000001FF  8F   CA   025A          BICL    #511,R2
                    06   11   0261          BRB     MAP_IT                          ; Noop to branch to ADDTRC for tracing.
          00000004'EF    17   0263          JMP     ADDTRC                          ;
                              0269  MAP_IT:
                    14   BB   0269          PUSHR   #^M<R2,R4>                      ; Save registers needed after algbpb.
                 FD92'   30   026B          BSBW    RM$ALGBPB                       ; Get Global Buffer Pointer Block.
                 B7 50   E9   026E          BLBC    R0, ALGBPERR                    ; Branch on error.
                 FD8C'   30   0271          BSBW    RM$ALGBPB                       ; Get Global Buffer Pointer Block.
                 B8 50   E9   0274          BLBC    R0, ALGBPERR1                   ; Branch on error.
                 FD86'   30   0277          BSBW    RM$ALBLB                        ; Get a BLB.
                 A4 50   E9   027A          BLBC    R0, ALBLBERR1                   ; Branch on error.
                 FD80'   30   027D          BSBW    RM$ALBLB                        ; Get a BLB.
                 97 50   E9   0280          BLBC    R0, ALBLBERR                    ; Exit on error.
                    14   BA   0283          POPR    #^M<R2,R4>                      ; Restore registers.
               0088 CA   D5   0285          TSTL    IFB$L_GBH_PTR(R10)              ; Already have gbl buffs?
                    07   13   0289          BEQL    1$                              ; No, then go on to map it.
                              028B          SSB     #IRB$V_GBLBUFF, (R9)            ; Note irab has extra gbpb, blb.
                  FF04   31   028F          BRW     EXIT                            ; Branch to exit.
                              0292
                              0292  ;
                              0292  ; R2 = Number of bytes to allocate (rounded up to full pages)
                              0292  ;
                              0292
                    7E   7C   0292  1$:     CLRQ    -(SP)                           ; Zero INADR forces P0 space to be allocated
                    7E   7C   0294          CLRQ    -(SP)                           ; Reserve space for RETADR.
                              0296
                              0296  ;
                              0296  ; The section name will be the ascii text '_RMS$' followed by the
                              0296  ; FCB address in hexadecimal.
                              0296  ;
                              0296
               5E   10   C2   0296          SUBL2   #16, SP                         ; Make room for gsd name.
                    6E   DF   0299          PUSHAL  (SP)                            ; Addr part of descriptor.
                    0D   DD   029B          PUSHL   #13                             ; Length of GSD name.
               020C 8F   BB   029D          PUSHR   #^M<R2,R3,R9>                   ; Save these around GETCCB call.
               59   5A   D0   02A1          MOVL    R10, R9                         ; Need ifab in r9.
                 FD59'   30   02A4          BSBW    RM$GETCCB                       ; Get CCB addr into R1.
               020C 8F   BA   02A7          POPR    #^M<R2,R3,R9>                   ; Restore registers.
          51   04   A1   D0   02AB          MOVL    CCB$L_WIND(R1), R1              ; Get ptr to window.
          51   18   A1   D0   02AF          MOVL    WCB$L_FCB(R1), R1               ; Get FCB addr into R1.
               FD4A CF   DF   02B3          PUSHAL  FAOCNTRL+1                      ;
     7E        FD45 CF   9A   02B7          MOVZBL  FAOCNTRL, -(SP)                 ; Build descriptor for control string.
          50        5E   D0   02BC          MOVL    SP, R0                          ; Need to pass addr of desc.
```

```
                         02BF          $FAO_S   CTRSTR=(R0),-            ; Address of control string descriptor
                         02BF                   OUTBUF=8(R0),-           ; Addr of output buffer descriptor.
                         02BF                   P1=R1                    ; FCB addr to show up in output string.
             6E    7C    02CF          CLRQ     (SP)                     ; Clear priv mask.
                         02D1          SSB      #PRV$V_SYSGBL, (SP)      ; Need sysgbl privilege.
       51    5E    D0    02D5          MOVL     SP, R1                   ; Save this stack address.
                         02D8          $SETPRV_S ENBFLG=#1,-             ; Turn on sysgbl for crmpsc.
                         02D8                   PRVADR=(R1),-
                         02D8                   PRVPRV=(R1)              ; Get previous state.
                         02E7
      51    08 AE  DE    02E7          MOVAL    8(SP), R1                ; Address of gsd name desc.
  50    52    17   9C    02EB          ROTL     #23, R2, R0              ; Get page count into r0.
                         02EF          $CRMPSC_S INADR = 32(R1),-        ; Point to array on stack.
                         02EF                   RETADR = 24(R1),-        ; Point to array on stack.
                         02EF                   GSDNAM = (R1),-          ;  ''                    ''
                         02EF                   PAGCNT = R0,-            ; Number of pages in section.
                         02EF                   ACMODE = #PSL$C_EXEC,-   ; Access mode is EXEC.
                         02EF          FLAGS = #SEC$M_GBL!SEC$M_SYSGBL!SEC$M_WRT!SEC$M_DZRO!SEC$M_PAGFIL!SEC$M_EXPR
                         0313
     1D 6E    19   E0    0313          BBS      #PRV$V_SYSGBL, (SP), 5$  ; If already had sysgbl, skip turnoff.
          6E    7C       0317          CLRQ     (SP)                     ; Init priv mask.
                         0319          SSB      #PRV$V_SYSGBL,(SP)       ; Turn off sysgbl.
       51    5E    D0    031D          MOVL     SP, R1                   ; Address of priv mask.
             50    DD    0320          PUSHL    R0                       ; Save status from crmpsc.
                         0322          $SETPRV_S PRVADR=(R1)             ; Turn off sysgbl.
             50  8ED0    0331          POPL     R0                       ; Restore crmpsc status.
                         0334   5$:
       5E    20    C0    0334          ADDL2    #32, SP                  ; Clean priv mask+name desc +name.
          06 50    E8    0337          BLBS     R0, 20$                  ; Continue if Ok.
          00ED    31     033A          BRW      SEC_ERR                  ; Branch to error code.
          00E3    31     033D   10$:   BRW      SEC_ERR1                 ; Branch to error code.
                         0340   20$:
51    04 AE   6E   C3    0340          SUBL3    (SP), 4(SP), R1          ; Get size allocated - 1.
             51    D6    0345          INCL     R1                       ; Size allocated.
       52    51    D1    0347          CMPL     R1, R2                   ; Get everything?
             F1    12    034A          BNEQ     10$                      ; Br if not.
       53    6E    D0    034C          MOVL     (SP),R3                  ; Move starting address of section into R3.
   50    0619 8F   B1    034F          CMPW     #SS$_CREATED, R0         ; Was the section just created?
             0B    13    0354          BEQL     30$                      ; Then it needs to be initialized.
08 A3    1611 8F   B1    0356          CMPW     #<GBH$C_BID+<GBH$C_BLN/4@8>>, GBH$B_BID(R3) ; Seem legit?
             DF    12    035C          BNEQ     10$                      ; NEQ there's an error.
          0087    31     035E          BRW      STORE_PTR                ; Else use it.
                         0361
                         0361   ;
                         0361   ; Initialize newly created section.
                         0361   ; R3 = start address of section
                         0361   ; R2 = size of section in bytes
                         0361   ; R6 = number of buffers in section.
                         0361   ;
                         0361
     0C A3    01   CE    0361   30$:   MNEGL    #1, GBH$L_HI_VBN(R3)     ; Store hi vbn for scan end check.
     10 A3    52   D0    0365          MOVL     R2, GBH$L_GS_SIZE(R3)    ; Store size of section in section.
     38 A4    52   D0    0369          MOVL     R2, GBSB$L_GS_SIZE(R4)   ; Store size of section in GBSB.
     34 A4    56   B0    036D          MOVW     R6, GBSB$W_GBC(R4)       ; Store number of buffers in section.
08 A3    1611 8F   B0    0371          MOVW     #<GBH$C_BID+<GBH$C_BLN/4@8>>, GBH$B_BID(R3) ; Store id, bln.
   50    0058 8F   3C    0377          MOVZWL   #GBH$C_BLN, R0           ; Offset to first GBD from GBH.
     04 A3    50   D0    037C          MOVL     R0, GBH$L_GBD_BLNK(R3)   ; Back link to GBD's.
```

```
            28 A3    50   D0  0380              MOVL    R0, GBH$L_GBD_START(R3) ; Save offset to first GBD.
            30 A3    50   D0  0384              MOVL    R0, GBH$L_GBD_NEXT(R3)  ; First GBD is first victim.
            34 A3    08   D0  0388              MOVL    #8, GBH$L_SCAN_NUM(R3)  ; Assume scan size of 8.
               56    08   D1  038C              CMPL    #8, R6                  ; Have at least 8 buffers?
               04    1B       038F              BLEQU   45$                     ; LEQU just use 8.
            34 A3    56   D0  0391              MOVL    R6, GBH$L_SCAN_NUM(R3)  ; Else only use # in section.
                     56   D7  0395      45$:    DECL    R6                      ; Num - 1.
               56    28   C4  0397              MULL2   #GBD$C_BLN, R6          ; Offset to last GBD from first.
 52    56  0000027F  8F   C1  039A              ADDL3   #GBD$C_BLN+GBH$C_BLN+511, R6, R2 ; End of GBD's + page-1 byte.
          52   01FF  8F   AA  03A2              BICW2   #511, R2                ; Round off to even page.
               50    53   C0  03A7              ADDL2   R3, R0                  ; Start address of GBD's.
               56    50   C0  03AA              ADDL2   R0, R6                  ; Addr of last GBD.
                              03AD              ASSUME  GBH$L_GBD_FLNK EQ 0
         63    56    53   C3  03AD              SUBL3   R3, R6, (R3)            ; Forw link points to last GBD.
            2C A3    63   D0  03B1              MOVL    (R3), GBH$L_GBD_END(R3) ; Offset to last GBD.
                              03B5      50$:
                              03B5              ASSUME  GBD$L_FLINK EQ 0
               60    28   CE  03B5              MNEGL   #GBD$C_BLN, (R0)        ; Offset to next GBD.
            04 A0    28   D0  03B8              MOVL    #GBD$C_BLN, GBD$L_BLINK(R0) ; Offset to last GBD.
                              03BC              ASSUME  GBD$B_BLN EQ <GBD$B_BID + 1>
      08 A0   0A13   8F   B0  03BC              MOVW    #<GBD$C_BID+<GBD$C_BLN/4@8>>, GBD$B_BID(R0) ; Id and bln.
         0C A0   01       CE  03C2              MNEGL   #1, GBD$L_VBN(R0)       ; Init VBN to -1.
         1A A0   55       B0  03C6              MOVW    R5, GBD$W_SIZE(R0)      ; Store buffer size.
         1C A0   52       D0  03CA              MOVL    R2, GBD$L_REL_ADDR(R0)  ; Store offset to buffer.
            52   55       C0  03CE              ADDL2   R5, R2                  ; Point to next buffer.
    FFDE 50   28   56     F1  03D1              ACBL    R6, #GBD$C_BLN, R0, 50$ ; Loop until past last GBD.
                              03D7              ASSUME  GBH$L_GBD_FLNK EQ 0
         04 A6   63       CE  03D7              MNEGL   (R3), GBD$L_BLINK(R6)   ; Last GBD's back link is
                              03DB                                              ; opposite of header's forw link.
            04 A3         CE  03DB              MNEGL   GBH$L_GBD_BLNK(R3),-    ; First GBD's forw link is
            58 A3             03DE                      GBH$C_BLN+GBD$L_FLINK(R3) ; opposite of header's back link.
                              03E0
                              03E0      ;
                              03E0      ; If tracing is to be enabled, noop the following branch.
                              03E0      ;
                              03E0
                 06   11      03E0              BRB     STORE_PTR               ; To make it easy to patch in tracing.
        00000028'EF   17      03E2              JMP     INIT_TRC                ; To init tracing blocks.
                              03E8      STORE_PTR:
            5E   10   C0      03E8              ADDL2   #16, SP                 ; 'Pop' INADR, RETADR arrays off stack.
              1C A3   D6      03EB              INCL    GBH$L_USECNT(R3)        ; Increment accessor count for section.
       0088 CA   53   D0      03EE              MOVL    R3, IFB$L_GBH_PTR(R10)  ; Point to the section.
                 59   D5      03F3              TSTL    R9                      ; Irab present?
                 04   13      03F5              BEQL    20$                     ; EQL then no irab.
                              03F7              SSB     #IRB$V_GBLBUFF, (R9)    ; Note this irab has extra gbpb, blb.
         1C A3   01   D1      03FB      20$:    CMPL    #1,GBH$L_USECNT(R3)     ; Are we first accessor?
                 16   12      03FF              BNEQ    30$                     ; No, branch to release lock.
            0080 CA   D0      0401              MOVL    IFB$L_PAR_LOCK_ID(R10),-; Save file lock id in global section.
              14 A3           0405                      GBH$L_LOCK_ID(R3)
              FBF6'   30      0407              BSBW    RM$LOWER_SYSLOCK        ; Turn file lock into system lock.
              FBF3'   30      040A              BSBW    RM$LOWER_GBS_LOCK       ; Lower lock on global buffer section.
        54    78 AA   D0      040D              MOVL    IFB$L_SFSB_PTR(R10),R4  ; Put address of SFSB in R4 for INIT_SFSB.
              FBEC'   30      0411              BSBW    RM$INIT_SFSB_IRB        ; Get a file lock for process using IRB to s
               FD7F   31      0414              BRW     EXIT                    ; Continue.
              14 A3   D0      0417      30$:    MOVL    GBH$L_LOCK_ID(R3),-     ; Move the parent lock id for bucket
            0080 CA           041A                      IFB$L_PAR_LOCK_ID(R10)  ;  locks into ifab from global buffer header
              FBE0'   30      041D              BSBW    RM$LOWER_GBS_LOCK       ; Do lock mode conversion.
```

```
               FD73   31  0420           BRW     EXIT                       ; Continue.
                          0423
                          0423      ;
                          0423      ; An error has been detected.  Disassociate from section, return structures
                          0423      ; already allocated.
                          0423      ;
                          0423
                          0423      SEC_ERR1:
       50  000184D4 8F   D0  0423           MOVL    #RMS$_DME, R0              ; Give DME error if not all mapped.
                          042A      SEC_ERR:
                          042A             SSUME   FAB$L_STV EQ RAB$L_STV
          0C A8    50   D0  042A           MOVL    R0, RAB$L_STV(R8)          ; Save error code.
                   50    8E   7D  042E           MOVQ    (SP)+, R0                  ; Get RETADR off stack into r0 and r1.
                   5E    08   C0  0431           ADDL2   #8, SP                     ; Pop INADR off stack.
                          0434           RMSERR  CRMP, -(SP)                ; Note error.
               FBC4'  30  0439           BSBW    RM$UNMAP_GBL_ALT           ; Delete the whole VA.
               FDAD   31  043C           BRW     ERL0                       ; Branch to finish up.
                          043F
```

```
                              043F            $NEWPSECT       RMSTRACE
              00000190        0000  NUMTRC: .LONG   400                     ; Number of trace blocks to allocate.
                              0004
                              0004  ;
                              0004  ; Add in extra bytes for trace blocks after size of section is determined.
                              0004  ;
                              0004
                              0004  ADDTRC:
         50   F9 AF    D0     0004          MOVL    NUMTRC, R0              ; Get number of trace blocks desired.
                 02    12     0008          BNEQ    10$                     ; Branch if non-zero.
                 50    D6     000A          INCL    R0                      ; Get at least one.
   50  00000040 8F    C4     000C  10$:    MULL2   #TRC$C_BLN, R0          ; Get size of trace blocks.
   50  000001FF 8F    C0     0013          ADDL2   #511, R0                ; Add in almost a page.
         50   01FF 8F    AA     001A          BICW2   #511, R0                ; Round to even page's worth.
              52   50    C0     001F          ADDL2   R0, R2                  ; Add in to size being requested.
           00000269'EF    17     0022          JMP     MAP_IT                  ; And return to mainline.
                              0028
                              0028  ;
                              0028  ; Initialize the trace blocks and pointer from the global buffer header.
                              0028  ;
                              0028  ; R3 - pointer to GBH
                              0028  ; R5 - buffer size
                              0028  ;
                              0028
                              0028  INIT_TRC:
                              0028          ASSUME  GBH$L_GBD_FLNK EQ 0
      50   53   63    C1     0028          ADDL3   (R3), R3, R0            ; Get address of last GBD in list.
   50   55   1C A0    C1     002C          ADDL3   GBD$L_REL_ADDR(R0), R5, R0 ; R0 now first byte after last buff.
  20 A3   50   20    C3     0031          SUBL3   #GBH$C_TRC_FLNK, R0, GBH$L_TRC_FLNK(R3) ; Offset to 1st trc blk
           50   53    C0     0036          ADDL2   R3, R0                  ; R0 now addr of first trace block.
   51   53   10 A3    C1     0039          ADDL3   GBH$L_GS_SIZE(R3), R3, R1 ; Get addr of end of gbl sec.
   51  00000040 8F    C2     003E          SUBL2   #TRC$C_BLN, R1          ; Limit for last trace block.
                              0045  10$:
                              0045          ASSUME  <TRC$C_BLN & 7> EQ 0    ; These will line up on quad boundary.
                              0045          ASSUME  TRC$L_FLNK EQ 0
   80  00000040 8F    D0     0045          MOVL    #TRC$C_BLN, (R0)+       ; Fwd offset to next block.
                              004C          ASSUME  TRC$L_BLNK EQ 4
   80  00000040 8F    CE     004C          MNEGL   #TRC$C_BLN, (R0)+       ; Back offset to last block.
                              0053          ASSUME  TRC$B_BID EQ 8
                              0053          ASSUME  TRC$B_BLN EQ <TRC$B_BID + 1>
      80   1012 8F    B0     0053          MOVW    #<TRC$C_BID+<TRC$C_BLN/4@8>>, (R0)+ ; Store id and bln.
 FFE7 50   36   51    F1     0058          ACBL    R1, #TRC$C_BLN-10, R0, 10$ ; Keep going until past limit.
                              005E
   50  00000040 8F    C2     005E          SUBL2   #TRC$C_BLN, R0          ; Back up to last trace block.
      51   50   53    C3     0065          SUBL3   R3, R0, R1              ; R1 is offset to last trc blk.
  24 A3   51   20    C3     0069          SUBL3   #GBH$L_TRC_FLNK, R1, GBH$L_TRC_BLNK(R3) ; Back link in header.
        60   24 A3    CE     006E          MNEGL   GBH$L_TRC_BLNK(R3),TRC$L_FLNK(R0) ; Flnk to hdr from last trc.
        50   20 A3    DE     0072          MOVAL   GBH$L_TRC_FLNK(R3), R0  ; Addr of flnk from header.
           50   60    C0     0076          ADDL2   (R0), R0                ; Get first trace block.
    04 A0   20 A3    CE     0079          MNEGL   GBH$L_TRC_FLNK(R3), TRC$L_BLNK(R0) ; Fix it's back link.
           000003E8'EF    17     007E          JMP     STORE_PTR               ; Jump back to main line.
                              0084
```

```
                                 0084  ;
                                 0084  ; Routine called to store information in trace block from initial call
                                 0084  ; to cache routine.
                                 0084  ;
                                 0084  ; AP is destroyed.  All other registers preserved.
                                 0084
                                 0084  RMSCACH_IN::
                       03   BB   0084          PUSHR   #^M<R0,R1>                 ; Save registers used.
           50    0088 CA   D0   0086  10$:    MOVL    IFB$L_GBH_PTR(R10), R0     ; Get pointer to gbh, if any.
                       44   13   008B          BEQL    EX2                        ; Exit if none.
                     0236   30   008D          BSBW    REMQT                      ; Get a trace block.
                       3F   13   0090          BEQL    EX2                        ; Exit if none.
             50    0A A0   9E   0092          MOVAB   TRC$W_FUNCTION(R0), R0     ; Get addr of function cell.
                  80    01   B0   0096          MOVW    #GBH$M_CACHE_IN, (R0)+     ; Note this function.
                  80    59   D0   0099          MOVL    R9, (R0)+                  ; structure
        5C   00000000'9F   D0   009C          MOVL    @#CTL$GL_PCB, AP           ; Get pcb addr.
             80    60 AC   B0   00A3          MOVW    PCB$L_PID(AP), (R0)+       ; pid
                     0219   30   00A7          BSBW    CNT                        ; seqnum
             80    04 AE   D0   00AA          MOVL    4(SP), (R0)+               ; vbn
             80    0C AE   D0   00AE          MOVL    12(SP), (R0)+              ; return1
             80    20 AE   D0   00B2          MOVL    32(SP), (R0)+              ; return2
                  80    53   D0   00B6          MOVL    R3, (R0)+                  ; arg_flg
                       80   D4   00B9          CLRL    (R0)+                      ; bdb_addr
                       80   7C   00BB          CLRQ    (R0)+                      ; not used
                       80   7C   00BD          CLRQ    (R0)+                      ; not used
                       80   7C   00BF          CLRQ    (R0)+                      ; not used
  51    50   00000040 8F   C3   00C1          SUBL3   #TRC$C_BLN, R0, R1         ; Get addr of trc blk
           50    0088 CA   D0   00C9          MOVL    IFB$L_GBH_PTR(R10), R0     ; Get addr of gbh.
                     01FF   30   00CE          BSBW    INSQH                      ; Insert blk at head of list.
                       03   BA   00D1  EX2:    POPR    #^M<R0,R1>                 ; Restore registers.
                       05        00D3          RSB                                ; Return to cache
                                 00D4
                                 00D4
```

```
                                0004  ;
                                0004  ; Store useful information from cache exit.
                                0004  ;
                                0004
                                0004  RMSCACH_OUT::
                 03       BB    0004          PUSHR   #^M<R0,R1>                ; Save registers.
        50     0088 CA    D0    00D6  1$:     MOVL    IFB$L_GBH_PTR(R10), R0    ; Get GBH ptr, if any.
                   F4     13    00DB          BEQL    EX2                       ; Exit if none.
                 01E6     30    00DD          BSBW    REMQT                     ; Remove a trc blk from tail.
                   EF     13    00E0          BEQL    EX2                       ; Exit if none.
        50       0A A0    9E    00E2          MOVAB   TRC$W_FUNCTION(R0), R0    ; Ptr to func field.
           80    02       B0    00E6          MOVW    #GBH$M_CACHE_OUT, (R0)+   ; function
           80    59       D0    00E9          MOVL    R9, (R0)+                 ; structure
    5C   00000000'9F      D0    00EC          MOVL    @#CTL$GL_PCB, AP          ; Addr of PCB
        80       60 AC    B0    00F3          MOVW    PCB$L_PID(AP), (R0)+      ; pid
                 01C9     30    00F7          BSBW    CNT                       ; structure
                   80     D4    00FA          CLRL    (R0)+                     ; vbn
        80       0C AE    D0    00FC          MOVL    12(SP), (R0)+             ; return1
        80       24 AE    D0    0100          MOVL    36(SP), (R0)+             ; return2
           80    6E       D0    0104          MOVL    (SP), (R0)+               ; arg_flg
           80    54       D0    0107          MOVL    R4, (R0)+                 ; bdb_addr
                   45     13    010A          BEQL    10$
     EC A0       1C A4    D0    010C          MOVL    BDB$L_VBN(R4), -20(R0)
        80       0C A4    B0    0111          MOVW    BDB$W_USERS(R4), (R0)+
        80       0E A4    B0    0115          MOVW    BDB$W_BUFF_ID(R4), (R0)+
        80       0B A4    90    0119          MOVB    BDB$B_CACHE_VAL(R4), (R0)+
        80       0A A4    90    011D          MOVB    BDB$B_FLGS(R4), (R0)+
        80       20 A4    D0    0121          MOVL    BDB$L_VBNSEQNO(R4), (R0)+
        51       10 A4    D0    0125          MOVL    BDB$L_BLB_PTR(R4), R1
                   2E     13    0129          BEQL    20$
        80       0B A1    90    012B          MOVB    BLB$B_MODEHELD(R1), (R0)+
        80       0A A1    90    012F          MOVB    BLB$B_BLBFLGS(R1), (R0)+
           80    51       D0    0133          MOVL    R1, (R0)+
        80       24 A1    D0    0136          MOVL    BLB$L_LOCK_ID(R1), (R0)+
        80       28 A1    D0    013A          MOVL    BLB$L_VALSEQNO(R1), (R0)+
                                013E  5$:
51    50   00000040 8F    C3    013E          SUBL3   #TRC$C_BLN, R0, R1        ; Get ptr to trc blk to insert.
        50     0088 CA    D0    0146          MOVL    IFB$L_GBH_PTR(R10), R0
                 0182     30    014B          BSBW    INSQH                     ; Insert at head of queue.
                 FF80     31    014E          BRW     EX2                       ; Branch to exit.
                   80     7C    0151  10$:    CLRQ    (R0)+
                   80     7C    0153          CLRQ    (R0)+
                   80     7C    0155  15$:    CLRQ    (R0)+
                   E5     11    0157          BRB     5$
                   80     B4    0159  20$:    CLRW    (R0)+
                   80     D4    015B          CLRL    (R0)+
                   F6     11    015D          BRB     15$
```

```
                              015F
                              015F  ;
                              015F  ; Store trace info for initial call to release.
                              015F  ;
                              015F  ;
                              015F  RMSRLS_IN::
                   07  BB  015F           PUSHR   #^M<R0,R1,R2>
     50    0088 CA  D0  0161  1$:         MOVL    IFB$L_GBH_PTR(R10), R0
                   03  12  0166           BNEQ    3$
                 00A3  31  0168           BRW     EX1
                 0158  30  016B  3$:      BSBW    REMQT                      ; Get trc blk from end.
                   03  12  016E           BNEQ    4$                         ; Branch if got one.
                 009B  31  0170           BRW     EX1                        ; Else exit.
     50      0A A0  9E  0173  4$:         MOVAB   TRC$W_FUNCTION(R0), R0
             80   04  B0  0177            MOVW    #GBH$M_RLS_IN, (R0)+       ; function
             80   59  D0  017A            MOVL    R9, (R0)+                  ; structure
   5C  00000000'9F  D0  017D              MOVL    @#CTL$GL_PCB, AP
     80      60 AC  B0  0184              MOVW    PCB$L_PID(AP), (R0)+       ; pid
                 0138  30  0188           BSBW    CNT                        ; seqnum
                           018B
                   51  7C  018B           CLRQ    R1
                   54  D5  018D           TSTL    R4
                   04  12  018F           BNEQ    5$
                           0191
                   80  D4  0191           CLRL    (R0)+                      ; VBN
                   20  11  0193           BRB     50$
                           0195  5$:
     08 A4      10  91  0195              CMPB    #BLB$C_BID, BLB$B_BID(R4)
                   05  12  0199           BNEQ    20$
          51      54  D0  019B            MOVL    R4, R1
                   07  11  019E           BRB     30$
     51      10 A4  D0  01A0  20$:        MOVL    BDB$L_BLB_PTR(R4), R1
          52      54  D0  01A4            MOVL    R4, R2
                   52  D5  01A7  30$:     TSTL    R2                         ; IS THERE BDB?
                   06  13  01A9           BEQL    40$
     80      1C A2  D0  01AB              MOVL    BDB$L_VBN(R2), (R0)+
                   04  11  01AF           BRB     50$
     80      14 A1  D0  01B1  40$:        MOVL    BLB$L_VBN(R1), (R0)+
     80      10 AE  D0  01B5  50$:        MOVL    16(SP), (R0)+              ; RETURN1
     80      20 AE  D0  01B9              MOVL    32(SP), (R0)+              ; RETURN2
          80      53  D0  01BD            MOVL    R3, (R0)+                  ; FLAGS
          80      52  D0  01C0            MOVL    R2, (R0)+                  ; BDB ADDR
                   16  13  01C3           BEQL    60$
     80      0C A2  B0  01C5              MOVW    BDB$W_USERS(R2), (R0)+
     80      0E A2  B0  01C9              MOVW    BDB$W_BUFF_ID(R2), (R0)+
     80      0B A2  90  01CD              MOVB    BDB$B_CACHE_VAL(R2), (R0)+
     80      0A A2  90  01D1              MOVB    BDB$B_FLGS(R2), (R0)+
     80      20 A2  D0  01D5              MOVL    BDB$L_VBNSEQNO(R2), (R0)+
                   04  11  01D9           BRB     70$
                           01DB  60$:
                   80  7C  01DB           CLRQ    (R0)+
                   80  B4  01DD           CLRW    (R0)+
                   51  D5  01DF  70$:     TSTL    R1                         ; IS THERE BLB?
                   15  13  01E1           BEQL    80$
     80      08 A1  90  01E3              MOVB    BLB$B_MODEHELD(R1), (R0)+
     80      0A A1  90  01E7              MOVB    BLB$B_BLBFLGS(R1), (R0)+
          80      51  D0  01EB            MOVL    R1, (R0)+
             80     24 A1    D0  01EE  1038          MOVL    BLB$L_LOCK_ID(R1), (R0)+
             80     28 A1    D0  01F2  1039          MOVL    BLB$L_VALSEQNO(R1), (R0)+
                       06    11  01F6  1040          BRB     90$
                                 01F8  1041 80$:
                       80    B4  01F8  1042          CLRW    (R0)+
                       80    D4  01FA  1043          CLRL    (R0)+
                       80    7C  01FC  1044          CLRQ    (R0)+
                                 01FE  1045 90$:
     51     50   00000040 8F    C3  01FE  1046          SUBL3   #TRC$C_BLN, R0, R1
             50     0088 CA    D0  0206  1047          MOVL    IFB$L_GBH_PTR(R10), R0
                     00C2    30  020B  1048          BSBW    INSQH                   ; Insert element at head of queue.
                       07    BA  020E  1049 EX1:     POPR    #^M<R0,R1,R2>
                       05        0210  1050          RSB
```

```
                          0211
                          0211  ;
                          0211  ; Store trace info at exit of release routine.
                          0211  ;
                          0211
                          0211  RM$RLS_OUT::
                 07   BB  0211          PUSHR   #^M<R0,R1,R2>
     50    0088  CA   D0  0213  1$:     MOVL    IFB$L_GBH_PTR(R10), R0
                 03   12  0218          BNEQ    3$
               FFF1   31  021A          BRW     EX1
               00A6   30  021D  3$:     BSBW    REMQT                   ; Get trc blk from end of queue.
                 03   12  0220          BNEQ    4$                      ; Br if got one
               FFE9   31  0222          BRW     EX1                     ; Else quit.
     50      0A  A0   9E  0225  4$:     MOVAB   TRC$W_FUNCTION(R0), R0
           80    08   B0  0229          MOVW    #GBH$M_RLS_OUT, (R0)+   ; function
           80    59   D0  022C          MOVL    R9, (R0)+               ; structure
5C      00000000'9F   D0  022F          MOVL    @#CTL$GL_PCB, AP
     80      60  AC   B0  0236          MOVW    PCB$L_PID(AP), (R0)+    ; pid
               0086   30  023A          BSBW    CNT                     ; seqnum
                          023D
                 51   7C  023D          CLRQ    R1
                 54   D5  023F          TSTL    R4
                 04   12  0241          BNEQ    5$
                          0243
                 80   D4  0243          CLRL    (R0)+                   ; VBN
                 20   11  0245          BRB     50$
                          0247  5$:
     08  A4      10   91  0247          CMPB    #BLB$C_BID, BLB$B_BID(R4)
                 05   12  024B          BNEQ    20$
           51    54   D0  024D          MOVL    R4, R1
                 07   11  0250          BRB     30$
     51      10  A4   D0  0252  20$:    MOVL    BDB$L_BLB_PTR(R4), R1
           52    54   D0  0256          MOVL    R4, R2
                 52   D5  0259  30$:    TSTL    R2                      ; IS THERE BDB?
                 06   13  025B          BEQL    40$
     80      1C  A2   D0  025D          MOVL    BDB$L_VBN(R2), (R0)+
                 04   11  0261          BRB     50$
     80      14  A1   D0  0263  40$:    MOVL    BLB$L_VBN(R1), (R0)+
     80      10  AE   D0  0267  50$:    MOVL    16(SP), (R0)+           ; RETURN1
     80      20  AE   D0  026B          MOVL    32(SP), (R0)+           ; RETURN2
           80    6E   D0  026F          MOVL    (SP), (R0)+             ; STATUS
           80    52   D0  0272          MOVL    R2, (R0)+               ; BDB ADDR
                 16   13  0275          BEQL    60$
     80      0C  A2   B0  0277          MOVW    BDB$W_USERS(R2), (R0)+
     80      0E  A2   B0  027B          MOVW    BDB$W_BUFF_ID(R2), (R0)+
     80      0B  A2   90  027F          MOVB    BDB$B_CACHE_VAL(R2), (R0)+
     80      0A  A2   90  0283          MOVB    BDB$B_FLGS(R2), (R0)+
     80      20  A2   D0  0287          MOVL    BDB$L_VBNSEQNO(R2), (R0)+
                 04   11  028B          BRB     70$
                          028D  60$:
                 80   7C  028D          CLRQ    (R0)+
                 80   B4  028F          CLRW    (R0)+
                 51   D5  0291  70$:    TSTL    R1                      ; IS THERE BLB?
                 15   13  0293          BEQL    80$
     80      08  A1   90  0295          MOVB    BLB$B_MODEHELD(R1), (R0)+
     80      0A  A1   90  0299          MOVB    BLB$B_BLBFLGS(R1), (R0)+
           80    51   D0  029D          MOVL    R1, (R0)+
                   80   24 A1   D0  02A0  1109          MOVL    BLB$L_LOCK_ID(R1), (R0)+
                   80   28 A1   D0  02A4  1110          MOVL    BLB$L_VALSEQNO(R1), (R0)+
                           06   11  02A8  1111          BRB     90$
                                    02AA  1112 80$:
                           80   B4  02AA  1113          CLRW    (R0)+
                           80   D4  02AC  1114          CLRL    (R0)+
                           80   7C  02AE  1115          CLRQ    (R0)+
                                    02B0  1116 90$:
      51    50   00000040 8F    C3  02B0  1117          SUBL3   #TRC$C_BLN, R0, R1
                  50   0088 CA  D0  02B8  1118          MOVL    IFB$L_GBH_PTR(R10), R0
                         0010   30  02BD  1119          BSBW    INSQH                         ; Insert at head of queue.
                         FF4B   31  02C0  1120          BRW     EX1
                                    02C3  1121
                                    02C3  1122 CNT:
                           80   B4  02C3  1123          CLRW    (R0)+
                                05  02C5  1124          RSB
                                    02C6  1125
                                    02C6  1126 ;CRASH: RMSPBUG -99
```

```
                              02C6   ;
                              02C6   ; Routine to remove an element from the end of a self relative queue.
                              02C6   ; The forward and back links in the removed element remain intact.
                              02C6   ;
                              02C6   ; Input: R0 - GBH header.
                              02C6   ; Output: R0 - trc blk element to use.
                              02C6   ; R1 destroyed.
                              02C6   ;
                              02C6
                              02C6 REMQT:
          20 A0     D5  02C6           TSTL    GBH$L_TRC_FLNK(R0)      ; Make sure trace blocks exits.
             04     13  02C9           BEQL    10$                     ; EQL there aren't any.
  50      20 A0     5F  02CB           REMQTI  GBH$L_TRC_FLNK(R0),R0   ; Remove a trc block from end of queue.
                    05  02CF 10$:      RSB                             ; Return.
                        02D0
                        02D0   ;
                        02D0   ; Routine to insert the trc blk previously removed from the tail of the queue
                        02D0   ; onto the head of the queue.
                        02D0   ;
                        02D0   ; Input:
                        02D0   ; R0 - GBH ptr.
                        02D0   ; R1 - element to insert.
                        02D0   ;
                        02D0
                        02D0 INSQH:
  20 A0     61      5C  02D0           INSQHI  (R1),GBH$L_TRC_FLNK(R0) ; Insert onto front of queue.
                    05  02D4           RSB                             ; And return.
                        02D5
                        02D5           $PSECT_RESTORE
                        043F           .END
```

```
                                                                    F  9
RM1CONN                          SEQUENTIAL AND COMMON CONNECT              16-SEP-1984 00:44:47  VAX/VMS Macro V04-00          Page  29
Symbol table                                                                 5-SEP-1984 16:23:11  [RMS.SRC]RM1CONN.MAR;1               (20)

$$.PSECT_EP                    = 00000000                 FAB$M_BIO                    = 00000020
$$RMSTEST                      = 0000001A                 FAB$V_BRO                    = 00000006
$$RMS_PBUGCHK                  = 00000010                 FAB$W_GBC                    = 00000048
$$RMS_TBUGCHK                  = 00000008                 FAOCNTRL                       00000000 R       01
$$RMS_UMODE                    = 00000004                 GBD$B_BID                    = 00000008
$$T1                           = 00000000                 GBD$B_BLN                    = 00000009
$$T2                           = 00000004                 GBD$C_BID                    = 00000013
ADDTRC                           00000004 R       03      GBD$C_BLN                    = 00000028
AGAIN                            00000169 R       01      GBD$L_BLINK                  = 00000004
ALBLBERR                         0000021A R       01      GBD$L_FLINK                  = 00000000
ALBLBERR1                        00000221 R       01      GBD$L_REL_ADDR               = 0000001C
ALGBPERR                         00000228 R       01      GBD$L_VBN                    = 0000000C
ALGBPERR1                        0000022F R       01      GBD$W_SIZE                   = 0000001A
ALLOC                            0000009D R       01      GBH$B_BID                    = 00000008
BAD_GBC                          00000236 R       01      GBH$C_BID                    = 00000011
BDB$B_CACHE_VAL                = 0000000B                 GBH$C_BLN                    = 00000058
BDB$B_FLGS                     = 0000000A                 GBH$L_GBD_BLNK               = 00000004
BDB$L_BLB_PTR                  = 00000010                 GBH$L_GBD_END                = 0000002C
BDB$L_VBN                      = 0000001C                 GBH$L_GBD_FLNK               = 00000000
BDB$L_VBNSEQNO                 = 00000020                 GBH$L_GBD_NEXT               = 00000030
BDB$W_BUFF_ID                  = 0000000E                 GBH$L_GBD_START              = 00000028
BDB$W_USERS                    = 0000000C                 GBH$L_GS_SIZE                = 00000010
BLB$B_BID                      = 00000008                 GBH$L_HI_VBN                 = 0000000C
BLB$B_BLBFLGS                  = 0000000A                 GBH$L_LOCK_ID                = 00000014
BLB$B_MODEHELD                 = 0000000B                 GBH$L_SCAN_NUM               = 00000034
BLB$C_BID                      = 00000010                 GBH$L_TRC_BLNK               = 00000024
BLB$L_LOCK_ID                  = 00000024                 GBH$L_TRC_FLNK               = 00000020
BLB$L_VALSEQNO                 = 00000028                 GBH$L_USECNT                 = 0000001C
BLB$L_VBN                      = 00000014                 GBH$M_CACHE_IN               = 00000001
BLKALC                           0000010D R       01      GBH$M_CACHE_OUT              = 00000002
CCB$L_WIND                     = 00000004                 GBH$M_RLS_IN                 = 00000004
CHKGBC                           000001B6 R       01      GBH$M_RLS_OUT                = 00000008
CHKMBC                           0000004A R       01      GBSB$L_GS_SIZE               = 00000038
CHK_GBC                          0000023D R       01      GBSB$W_GBC                   = 00000034
CLNT                             00000036 R       01      GIVEBACK                       000001A9 R       01
CNT                              000002C3 R       03      IFB$B_FAC                    = 00000022
CTL$GL_PCB                       ********   X     03      IFB$L_BDB_BLNK               = 00000044
DECR_BCNT                        00000194 R       01      IFB$L_BLBBLNK                = 0000009C
DEV$V_FOD                      = 0000000E                 IFB$L_DEVBUFSIZ              = 00000048
DEV$V_FOR                      = 00000018                 IFB$L_EBK                    = 00000074
DEV$V_REC                      = 00000000                 IFB$L_GBH_PTR                = 00000088
DEV$V_RND                      = 0000001C                 IFB$L_GBSB_PTR               = 0000007C
DEV$V_RTM                      = 0000001D                 IFB$L_PAR_LOCK_ID            = 00000080
DEV$V_SQD                      = 00000005                 IFB$L_PRIM_DEV               = 00000000
DEV$V_TRM                      = 00000002                 IFB$L_SFSB_PTR               = 00000078
ERL0                             000001EC R       01      IFB$V_BIO                    = 00000005
ERL1                             000001F4 R       01      IFB$V_DAP                    = 0000003E
ERL2                             0000020A R       01      IFB$V_EOF                    = 00000021
ERL3                             00000203 R       01      IFB$V_NORECLK                = 00000033
ERL4                             000001FC R       01      IFB$V_PPF_INPUT              = 0000002E
ERRDEV                           0000003C R       01      IFB$V_TEF                    = 00000036
ERRMBC                           00000043 R       01      IFB$W_AVLCL                  = 00000084
ERRRFM                           00000031 R       01      IFB$W_FFB                    = 0000005C
EX1                              0000020E R       03      IMP$V_IIOS                   = 00000000
EX2                              000000D1 R       03      IMP$V_NOPOBUFS               = 00000005
EXIT                             00000196 R       01      INIT_TRC                       00000028 R       03
FAB$L_STV                      = 0000000C                 INSQH                          000002D0 R       03
```

```
IRB$B_BCNT                      = 00000054               RMS$_MBC                        = 00018734
IRB$B_MBC                       = 00000055               RMS$_RFM                        = 00018664
IRB$B_MBF                       = 0000005C               ROP                             = 00000020
IRB$L_NRP_VBN                   = 00000040               SEC$M_DZRO                      = 00000004
IRB$L_NXTBDB                    = 0000003C               SEC$M_EXPREG                    = 00020000
IRB$V_EOF                       = 00000021               SEC$M_GBL                       = 00000001
IRB$V_GBLBUFF                   = 00000036               SEC$M_PAGFIL                    = 00080000
IRB$V_RAHWBH                    = 0000002A               SEC$M_SYSGBL                    = 00008000
IRB$W_NRP_OFF                   = 00000044               SEC$M_WRT                       = 00000008
MAP_IT                            00000269 R      01     SEC_ERR                           0000042A R      01
NT$CONNECT                        ********    X   01     SEC_ERR1                          00000423 R      01
NUMTRC                            00000000 R      03     SETNXT                            000000BE R      01
PCB$L_PID                       = 00000060               SS$_CREATED                     = 00000619
PIO$GB_DFMBC                      ********    X   01     STORE_PTR                         000003E8 R      01
PIO$GB_DFMBFSDK                   ********    X   01     SYS$CRMPSC                        ********   GX   01
PRV$V_SYSGBL                    = 00000019               SYS$FAO                           ********    X   01
PSL$C_EXEC                      = 00000001               SYS$GB_DFMBC                      ********    X   01
RAB$B_MBC                       = 00000037               SYS$GB_DFMBFSDK                   ********    X   01
RAB$B_MBF                       = 00000036               SYS$SETPRV                        ********   GX   01
RAB$B_ROP1                      = 00000005               TRC$B_BID                       = 00000008
RAB$L_FAB                       = 0000003C               TRC$B_BLN                       = 00000009
RAB$L_ROP                       = 00000004               TRC$C_BID                       = 00000012
RAB$L_STV                       = 0000000C               TRC$C_BLN                       = 00000040
RAB$M_RAH                       = 00000200               TRC$L_BLNK                      = 00000004
RAB$M_WBH                       = 00000400               TRC$L_FLNK                      = 00000000
RAB$V_BIO                       = 0000000B               TRC$W_FUNCTION                  = 0000000A
RAB$V_EOF                       = 00000008               UNIT                              000000A6 R      01
REMQT                             000002C6 R      03     WCB$L_FCB                       = 00000018
RM$ALBLB                          ********    X   01
RM$ALDBUF                         ********    X   01
RM$ALGBPB                         ********    X   01
RM$BDBALLOC                       00000111 RG     01
RM$BDBALLOC_ALT                   0000015E RG     01
RM$CACH_IN                        00000084 RG     03
RM$CACH_OUT                       000000D4 RG     03
RM$CCLNT                          ********    X   01
RM$COMCLNUP                       ********    X   01
RM$CONNECT1                       00000009 RG     01
RM$CONN_ERRRFM                    00000031 RG     01
RM$EXSUC                          ********    X   01
RM$EX_NOSTR                       ********    X   01
RM$GETCCB                         ********    X   01
RM$INIT_GBSB                      ********    X   01
RM$INIT_SFSB_IRB                  ********    X   01
RM$LOWER_GBS_LOCK                 ********    X   01
RM$LOWER_SYSLOCK                  ********    X   01
RM$RETBDB                         ********    X   01
RM$RETBLB                         ********    X   01
RM$RETGBPB                        ********    X   01
RM$RLS_GBSB                       ********    X   01
RM$RLS_IN                         0000015F RG     03
RM$RLS_OUT                        00000211 RG     03
RM$UNMAP_GBL_ALT                  ********    X   01
RMS$_CRMP                       = 0001C14C
RMS$_DEV                        = 000184C4
RMS$_DME                        = 000184D4
RMS$_GBC                        = 000187CC
```

```
RM1CONN                          SEQUENTIAL AND COMMON CONNECT        16-SEP-1984 00:44:47  VAX/VMS Macro V04-00         Page  31
Psect synopsis                                                         5-SEP-1984 16:23:11  [RMS.SRC]RM1CONN.MAR;1             (20)

                                                 +----------------+
                                                 ! Psect synopsis !
                                                 +----------------+

PSECT name                       Allocation             PSECT No.  Attributes
----------                       ----------             ---------  ----------
.  ABS  .                        00000000  (     0.)    00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
RM$RMS1                          0000043F  (  1087.)    01 (  1.)    PIC   USR   CON   REL   GBL NOSHR   EXE   RD   NOWRT NOVEC BYTE
$ABS$                            00000000  (     0.)    02 (  2.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
RM$TRACE                         000002D5  (   725.)    03 (  3.)    PIC   USR   CON   REL   GBL NOSHR   EXE   RD   NOWRT NOVEC BYTE

                                               +------------------------+
                                               ! Performance indicators !
                                               +------------------------+

Phase                     Page faults    CPU Time       Elapsed Time
-----                     -----------    --------       ------------
Initialization                     36    00:00:00.08    00:00:00.40
Command processing                137    00:00:00.68    00:00:04.36
Pass 1                            608    00:00:26.33    00:01:09.37
Symbol table sort                   0    00:00:03.90    00:00:04.89
Pass 2                            219    00:00:05.36    00:00:13.14
Symbol table output                32    00:00:00.22    00:00:00.74
Psect synopsis output               2    00:00:00.02    00:00:00.02
Cross-reference output              0    00:00:00.00    00:00:00.00
Assembler run totals             1036    00:00:36.60    00:01:32.94

The working set limit was 1950 pages.
145709 bytes (285 pages) of virtual memory were used to buffer the intermediate code.
There were 140 pages of symbol table space allocated to hold 2617 non-local and 71 local symbols.
1157 source lines were read in Pass 1, producing 20 object records in Pass 2.
45 pages of virtual memory were used to define 43 macros.

                                              +--------------------------+
                                              ! Macro library statistics !
                                              +--------------------------+

Macro library name                          Macros defined
------------------                          --------------
_$255$DUA28:[RMS.OBJ]RMS.MLB;1                   19
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                    4
_$255$DUA28:[SYSLIB]STARLET.MLB;2                16
TOTALS (all libraries)                           39

2809 GETS were required to define 39 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:RM1CONN/OBJ=OBJ$:RM1CONN MSRC$:RM1CONN/UPDATE=(ENH$:RM1CONN)+EXECML$/LIB+LIB$:RMS/LIB
```

0321 AH-BT13A-SE DIGITAL EQUIPMENT CORPORATION
VAX/VMS V4.0 CONFIDENTIAL AND PROPRIETARY